Discrete Spider Monkey Optimization for Travelling Salesman Problem

Discrete Spider Monkey Optimization for Traveling Salesman Problem

M. A. H. Akhand[1], Safial Islam Ayon[1], S. A. Shahriyar[1], N. Siddique[2] and H. Adeli[3]

[1] Dept. of CSE, Khulna University of Engineering & Technology, Khulna, Bangladesh
[2]School of Computing, Engineering and Intelligent Systems, Ulster University, United Kingdom
[3]The Ohio State University, USA

***Abstract*—** Meta-heuristic algorithms inspired by biological species have become very popular in recent years. Collective intelligence of various social insects such as ants, bees, wasps, termites, birds, fish, has been investigated to develop a number of meta-heuristic algorithms in the general domain of swarm intelligence (SI). The developed SI algorithms are found effective in solving different optimization tasks. Traveling Salesman Problem (TSP) is the combinatorial optimization problem where a salesman starting from a home city travels all the other cities and returns to home city in the shortest possible path. TSP is a popular problem due to the fact that the instances of TSP can be applied to solve real-world problems, implication of which turns TSP into a standard test bench for performance evaluation of new algorithms. Spider Monkey Optimization (SMO) is a recent addition to SI algorithms based on the social behavior of spider monkeys. SMO implicitly adopts grouping and regrouping for the interactions to improve solutions; such multi-population approach is the motivation of this study to develop an effective method for TSP. This paper presents an effective variant of SMO to solve TSP called discrete SMO (DSMO). In DSMO, every spider monkey represents a TSP solution where Swap Sequence (SS) and Swap Operator (SO) based operations are employed, which enables interaction among monkeys in obtaining the optimal TSP solution. The SOs are generated using the experience of a specific spider monkey as well as the experience of other members (local leader, global leader, or a randomly selected spider monkey) of the group. The performance and effectiveness of the proposed method have been verified on a large set of TSP instances and the outcomes are compared to other well-known methods. Experimental results demonstrate the effectiveness of the proposed DSMO for solving TSP.

***Index Terms*—** Traveling Salesman Problem, Swap Sequence, Swap Operator, Partial Search, Spider Monkey Optimization

1. INTRODUCTION

Meta-heuristic algorithms inspired by biological species have become very popular in the recent years. Bio-inspired algorithms are based on behaviours of some animals, insect societies and movement of organisms. Evolutionary algorithms are the early methods in 1960s inspired by biological evolution which has found wide spread applications in almost all branches of science and engineering [1]–[3]. In the last few decades, collective intelligence of various social insects such as ants, bees, wasps, termites, birds, fish, has been investigated towards the development of a number of meta-heuristic algorithms in the domain of swarm intelligence (SI) [4]–[7].

SI-based meta-heuristic algorithms became known with the advent of Particle Swarm Optimization (PSO) and Ant colony optimization (ACO) in the 1990s. PSO [8] is based on the social behaviour of birds flocking and fish schooling. Due to its simplicity, PSO has been successfully applied to many optimisation problems with modified and hybrid variants of PSO [9]–[12]. Ants are social insects and live in colonies. Inspiration of ACO [13], [14] is based on the foraging behaviour of ant colonies and it has been well-studied for different applications. Due to the enormous success of PSO and ACO, researchers explored swarm behaviours of other biological species for developing new algorithms. Among them are bee colony optimization [15][16] based on the foraging behaviour of bees and waggle dance communicating food source to the colony[17], firefly algorithm[18] based on the short and rhythmic flashing used for signalling to attract each other, glowworms swarm optimization [19] simulating the movement of the glowworms based on the distance between them and the luminescent quantity, and bat algorithm [20] based on echolocation behaviour of bats. Flower pollination algorithm[21], [22] based on biological evolution of pollination of the flowers, bacterial colony foraging [23][24] based on biomimicry of foraging bacteria, and salp swarm algorithm [25] enthused from salp navigating and foraging behaviour in oceans are inspirations from the microscopic organisms. Recently, living in groups and hunting behaviours of animals have been used to develop clever SI-based optimization techniques such as grey wolf optimization [26], group search optimizer [27], and spider monkey optimization [28]. Moreover, hybridization of strategies from two or more individual methods is also a new way of developing new meta-heuristic algorithm [29].

The SI-based algorithms are found effective in solving optimization tasks. These algorithms are grouped into two categories: numerical and combinatorial. In numerical optimization, an algorithm searches in a defined functional space to find optimal numerical values of different parameters of a function. In contrast, finding optimal combination of different discrete events satisfying different constraints is the task of combinatorial optimization. Traveling salesman problem (TSP) is the most studied combinatorial optimization problem due to its significant implications to various real-world applications such as scheduling, vehicle routing, placement of goods in warehouse, facility layout design in factory, and printed circuit design [30]. From optimization point of view, TSP is to find the shortest tour for a salesman travelling all cities and returning home city without visiting a city twice. ACO is one of the earliest SI-based meta-heuristic algorithms applied to TSP. Various numerical optimization problems are also verified on TSP with variations and/or modifications with success [31]–[33]. Interest has grown in the recent years for developing new algorithms and approaches to TSP considering it as a general test bench.

Among the existing SI methods, spider monkey optimization (SMO) has some considerable features over the others, which merit investigation for developing new optimization method for TSP. Spider monkeys search for food in a group under a leadership and share their information with each other [34], [35]. The group divides into subgroups when spider monkeys fail to find any food. The subgroups also have individual leaders and set for searching food in different directions. At some stage, all the subgroups are combined into one large main group. The social

behaviour of spider monkeys is an example of a Fission-Fusion Social Structure [36] where group members benefit from living together and sharing information. Inspired by this social behaviour of spider monkeys, Bansal et al. [28] proposed the SMO algorithm for numerical optimization. The subgrouping phenomenon is the distinct feature of SMO to show good performance for numerical problem and such approach might be well suited for TSP. Although SMO conceived basic features of spider monkeys for numerical optimization, it is completely different and more difficult to adapt for discrete optimization problems like TSP. The proposed discrete version of SMO in this study can accomplish the challenges of applying SMO to TSP by introducing a number of new operators to SMO.

In the proposed discrete SMO (DSMO), every spider monkey represents a TSP solution; and Swap Sequence (SS) and Swap Operator (SO)-based operations are considered for interaction among monkeys to find the optimal TSP solution. An SS is a group of SOs. Each contains a pair of indexes on the TSP tour. A new tour is generated transforming a TSP tour by swapping cities indicated by SOs of an SS. The SS generation to update a monkey for an improved solution is the key feature of the method and SS of a particular spider monkey is generated with interaction with other members of the group. In the case of updating a solution of spider monkey with generated SS, a Partial Search (PS) technique is deployed to achieve the best outcome with full or partial SS. The proposed method has been tested on a suite of benchmark TSPs and is proved to outperform other well-known metaheuristic methods applied to TSP.

The rest of the paper is organized as follows. Section 2 describes the standard SMO algorithm and its features. Section 3 provides the description of the proposed DSMO algorithm for TSP. Section 4 presents the experimental results of the proposed DSMO and compares the performance with other contemporary metaheuristic algorithms. Finally, Section 5 presents some concluding remarks out of this study.

2. REVIEW OF SPIDER MONKEY OPTIMIZATION (SMO)

SMO [28] is an SI algorithm for numerical optimization based on the social behavior of spider monkeys. Spider monkey is a South American species of monkey which lives in a large group and shows intelligence in social behavior and foraging [34], [35]. Spider monkeys search for food in a group (under a group leader) and subgroups (under subgroup leaders). They foraging for food starts with a single group and is divided into subgroups to spread the search into different directions for exploring the region better. A subgroup leader may divide the group again if failed to find food source [36]. When the number of groups reaches a maximum number, the main group leader combines all the subgroups into a main large group and then splits again into some subgroups. The process of fission and fusion is repeated until the swarm of spider monkeys ends up with a good food source. The searching mechanism is employed in SMO for numeral optimization. Like other SI algorithms, each monkey of SMO is a position in a multi-dimensional search space that represents a solution of the problem at hand. SMO starts with a population of random initial positions of the monkeys in the swarm. The positions are updated

Algorithm 1: SMO Algorithm

Step 1: Initialization
Step 2: Select **Local Leader** and **Global Leader**
Step 3: For each Spider Monkey
a. Generate new position // Local-Leader phase
b. Calculate the probability // Global-Leader phase
c. Update position again based on the probability
d. Select new **Local Leader** and **Global Leader**
e. Redirect all the members of the group if **Local Leader** is not updating for a specific time.
f. Splits the group into subgroups or combine all the groups into one group if **Global Leader** is not updating for a specific time.
Step 4: Goto **Step 3** if termination condition is not met
Step 5: Return the **Global Leader** as the final solution

in every iteration through interactions among the monkeys. The best position of all the monkeys is called the Global Leader. The best solution for each group is called the Local Leader of that group.

There are six phases in SMO namely Local-Leader phase, Global-Leader phase, Local-Leader-Learning phase, Global-Leader-Learning phase, Local-Leader-Decision phase and Global-Leader-Decision phase. Each phase has its own and unique purpose [37] which is described in Algorithm 1 in five steps. Local-Leader phase is responsible for producing a newly updated position for every spider monkey in the group (Step 3.a). In the Global-Leader phase, first a probability is calculated (Step 3.b), then the positions of all the spider monkeys are updated (Step 3.c). If the new position is better than the previous position, the position is updated to the new position otherwise it retains the current position. Local Leaders and Global Leader are selected in the Local-Leader-Learning phase and Global-Leader-Learning phase respectively (Step 3.d). In the Local-Leader-Decision phase and Global-Leader-Decision phase, the Local-Leaders initialize all the members of the groups again (Step 3.e) and the Global-Leader takes the decision to divide the group into subgroups or combine (i.e. fuse) all the groups into one group (Step 3.f).

SMO follows self-organization as well as division of labour properties for finding swarming activities of the animals. It demonstrates positive feedback mechanisms of self-organization because a spider monkey updates its position using Local Leader, Global Leader, and self-experience. When the Global Leader divides the group into subgroups, it represents the division of labour property observed in many species [36]. SMO considers two important control parameters: Global-Leader-Limit and Local-Leader-Limit, which help Local Leader and Global Leader to make the right decision. Finally, the Global Leader will provide the final outcome (i.e. solution) of SMO.

SMO has found a good number of applications in different domains, especially real valued optimization tasks, within a short period of time [38]. It has been applied to automatic generation of control [39], designing electromagnetic antenna arrays [40], rule mining for

diabetes classifications [41], designing optimal fuzzy rule-base for a Tagaki-Sugeno-Kang (TSK) fuzzy control system [42], improving quality and diversity of particles and distributing them in particle filters to provide a robust object tracking framework [43], CDMA multiuser detection [44], optical power flow, pattern synthesis of sparse linear array and antenna arrays [45], numerical classification [46], optimizing frequency in microgrid [47], economic dispatch problem [48], optimizing models of multi-reservoir system [49], and energy efficient clustering for WSNs [50].

A number of studies on modified SMO for optimization problems in diverse domains have also been reported in the literature. Some of the SMO algorithms are modified to solve global optimization problems [51], to improve the local search capability of algorithm (i.e., exploitation of the search space) capability of algorithm [52], [53] and proposed fitness-based position update strategy for the spider monkeys [54], modified ageist SMO incorporating age of the monkeys that further divides the groups of monkeys into subgroups according to the age based on different levels of ability [55], binary SMO [45], modified SMO for constraint continuous optimization [37], and modified SMO incorporating Nelder–Mead method to enhance local search [51].

Grouping method makes full use of the population information generated in optimization procedure and the regrouping strategy is equivalent to the resumption of evolutionary process[56][57][58]. Algorithms adapting grouping and regrouping mechanisms are found effective for different numerical optimizations [22], [59], [60]. In this line, SMO implicitly adopts grouping and regrouping for the interactions to improve solutions; such multi-population approach might be effective for discrete optimization (e.g., TSP). All the applications of SMO mentioned in the previous section are floating-point numeral optimization where mathematical operations (e.g., additional, subtraction, multiplication and division) are easy to implement for interaction within elements and individuals in SMO. On the other hand, solving discrete-type optimization tasks such as TSP inherits its own difficulties. Operators of standard SMO simply cannot be implemented for TSP or similar problems. New operators have to be designed or SMO has to be modified for new applications. Therefore, the main motivation of this study is to devise a mechanism such that discrete problems like TSP can be solved using SMO and hence this paper proposes a discrete SMO for TSP.

3. Discrete Spider Monkey Optimization (DSMO) for TSP

This section expounds the proposed DSMO and its application to TSP. In DSMO, every spider monkey represents a TSP solution; and Swap Sequence and Swap Operators based operations are developed for interaction among monkeys to find an optimal TSP solution. In this section, the operators are explained in detail first and then the DSMO method is presented for TSP employing the operators in the sequel.

3.1 Traveling Salesman Problem (TSP) and Its Importance

The problem was formulated in the nineteenth century but became known as TSP and a benchmark problem for combinatorial optimization problem in the 1950s and 1960s [61]. The TSP can be formulated as an undirected weighted graph denoted as $G = (V, E)$, where V is the set of cities (or nodes), and E is the path between cities. In TSP, a graph of N cities (i.e., a map) is given. The salesman is required to visit every city only once starting from an initial city and returning to the starting city. For a given graph with N cities, there are $(N-1)!/2$ possible solutions. Formally, if there are N cities, the TSP searches for a permutation $\pi: \{0, \cdots, N-1\} \rightarrow \{0, \cdots, N-1\}$ using a cost matrix $C = [c_{ij}]$, where c_{ij} is the cost of travel between city i and j, with the goal of minimizing the total path length:

$$f(\pi, c) = \sum_{i=0}^{N-1} c_{\pi(i),\ \pi(i+1)}, \tag{1}$$

where $\pi(i)$ is the city at i^{th} location in the tour. The goal of the TSP is to find a tour with the minimum total path length.

Real-world problems such as scheduling [62]–[66], routing [67], [68], printed circuit board design [12], electrical and telecommunication networks can be formulated as an instance of the TSP. TSP demonstrates all the features of combinatorial optimization problem and proved to be one of the NP-hard problems having exponential time complexity [69]. Therefore, TSP is being used by the research community of combinatorial optimization [61]. TSP is also considered as a test benchmark for performance verification of new metaheuristic algorithms for optimization.

Nature-inspired computing [70], [71] and metaheuristic algorithms are well studied for solving TSP. Genetic Algorithm (GA) is among the first of its kind applied to TSP in 1985 by Grefenstette et al. [72] and later on in the 1990s by a number of other researchers [73], [74]. In these applications, TSP is represented as a permutation problem where the crossover operation is straightforward and the mutation operation is carried out as a swap operation. Simulated annealing (SA) [75] has been applied to TSP as early as mid-1980s [76], [77]. Geng et al. [78] applied an adaptive SA to TSP. Electromagnetism optimization (EMO) algorithm [79] has been applied to TSP [80] demonstrating competitive performance. Gravitational Search Algorithm (GSA) [81], [82] has been applied to benchmark instances of TSP [83] where TSP is formulated as a permutation problem. Various other numerical optimization methods are also applied to TSP such as integer programming and dynamic programming [84], [85].

The interest grew among researchers in developing new SI algorithms based on insect's social behavior in solving TSP [86]. Ant Colony Optimization (ACO) has provided reasonable and quality solutions for TSP [14], [87]–[89]. A number of Particle Swarm Optimization (PSO)-based methods have been applied to TSP with success [90]–[92]. Artificial Bee Colony (ABC) was also applied to TSP by a number of researchers with considerable success [32], [93].

3.2 Swap Sequence and Swap Operator in TSP

Swap Sequence based operations are common in different prominent methods for solving TSP. A Swap Sequence (SS) having several Swap Operators (SOs) is considered to transform a TSP solution into a new solution [91], [92]. A TSP solution is a sequence of cities to be visited and an SO is a pair of position indexes which imply two cities to be interchanged (i.e., swapped) in the TSP solution. Assume the solution of a TSP of four cities in a sequence as $T = (4 - 1 - 3 - 2)$; and the modified solution S' applying an SO defined by $SO(1,3)$ is

$$T' = (\mathbf{4} - 1 - \mathbf{3} - 2) + SO(1,3) = (\mathbf{3} - 1 - \mathbf{4} - 2),$$

here '+' is not an arithmetic addition operation rather application of SO operation onto the solution. Swap Operator is an important factor to update TSP solution in this study and its operation is comparable to mutation operator used in GA.

An SS [91], [92] is a set of single or more SOs that is applicable to a specific TSP solution in a sequence expressed by:

$$SS = (SO_1, SO_2, SO_3, \cdots, SO_n), \tag{2}$$

where $SO_1, SO_2, SO_3, \cdots, SO_n$ are SOs. Finally, SS implication on a tour T_1 produces a new tour T_2. The implication is formulated as:

$$T_2 = T_1 + SS$$

$$T_2 = T_1 + (SO_1, SO_2, SO_3, \cdots, SO_n)$$

$$T_2 = [(((T_1 + SO_1) + SO_2) + SO_3) + \cdots) + SO_n] \tag{3}$$

The implication order of SOs in the SS is maintained as the implication of the same SOs in different orders on a solution may yield different new solutions. Another interpretation of swap sequence is the difference between two TSP solutions where *SS* may be calculated rearranging Eq. (3) as Eq. (4).

$$SS = T_2 - T_1 = (SO_1, SO_2, SO_3, \cdots, SO_n) \tag{4}$$

Here '-' is not an arithmetic subtraction operation rather it means the required SOs in the SS to transform T_1 into T_2. Suppose two solutions are T_1 = (1 – 2 – 3 – 4 – 5) and T_2 = (2 – 3 – 1 – 5 – 4) then the SS = *SO* (1,2), *SO* (2,3), *SO* (4,5).

Moreover, the operator ⊗ is considered to merge several SSs into a new SS. Suppose SS_1 = {*SO* (1,3), *SO* (4,3)} and SS_2 = {*SO* (3,1), *SO* (5,2)}. The new Swap Sequence SS' merging SS_1 and SS_2 will contain the SOs of both SSs.

$$\begin{aligned} SS' &= SS_1 \otimes SS_2 \\ &= \{SO\ (1,3),\ SO\ (4,3)\} \otimes \{SO\ (3,1),\ SO\ (5,2)\} \\ &= SO\ (1,3),\ SO\ (4,3),\ SO\ (3,1),\ SO\ (5,2) \end{aligned} \tag{5}$$

It is also to be noted that same result may be obtained using various SSs on a TSP solution. Among all these SSs, the SS which holds the minimum SOs is termed as the Basic Swap Sequence (BSS). Suppose two SSs are: SS_1 = {*SO* (3,2), *SO* (2,3), *SO* (1,4), *SO* (3,5), *SO* (4,5)} and SS_2 = {*SO* (1,4), *SO* (3,5), *SO* (5,4)}, which are applied to T_1 = (5-1-2-3-4) independently, yielding the outcome T_2 = (3-1-4-2-5). Consequently, SS_2 is the BSS which also is found employing $T_2 - T_1$ using Eq. (4).

Recently, Partial Search (PS) based SS operations are found efficient in solving TSP [31], [33]. Since each and every SO implication gives individual TSP solutions, PS technique considers each one as tentative tour and returns the best one as the outcome. Suppose a solution is T = (1 – 3 – 2 – 5 – 4) and SS = {*SO* (2,3), *SO* (1,2), *SO* (4,5)}. Using the three SOs, the three tentative tours (T_1, T_2 and T_3) and their corresponding TSP costs will be as follows:

Applying SS		Tentative TSP Tours	TSP Tour Cost
T + *SO (2,3)*	=>	T_1 = (1 – 2 – 3 – 5 – 4)	$Cost_1 = f(T_1)$
T_1 + *SO (1,2)*	=>	T_2 = (2 – 1 – 3 – 5 – 4)	$Cost_2 = f(T_2)$
T_2 + *SO (4,5)*	=>	T_3 = (2 – 1 – 3 – 4 – 5)	$Cost_3 = f(T_3)$

All three tentative tours are feasible TSP solutions and an intermediate tentative solution can be better than the last one. Thus, PS returns the solution having the lowest TSP cost [90]. In this study PS technique is considered to achieve the best outcome with full or partial SS while updating a solution of spider monkey with generated SS.

3.3 DSMO Approach to Solve TSP

The proposed DSMO method consists of multiple steps; among them the significant steps are: selection of Local Leaders and Global Leader, update of individual spider monkeys based on both types of leaders, updating the Local Leaders and Global Leader and decision phase of Local Leaders and Global Leader. Moreover, similar to other SI methods, initialization and termination steps are available in the proposed method. In the initialization step, a population of spider monkeys are defined with random tours; divide them into group(s); and select Local Leaders and a Global Leader. In every iteration, significant DSMO steps are followed and termination criteria is checked. The steps of DSMO are explained briefly in the following subsections.

1) Initialization

In the beginning, a population of N spider monkeys (SMs) is initialized with random TSP tours and fitness of those are measured. The population may be divided into n groups and the Local Leaders (LLs) of individual groups are selected based on the fitness. Among the LLs, the best one is chosen as the Global Leader (GL). DSMO starts with a single group having all the SMs. In such a case, LL and GL both are the same SM. There are four control parameters which are initialized at this stage: the Maximum number of Groups (*MG*) to be allowed,

perturbation rate (*pr*), *Local Leader Limit* (*LLL*), *Global Leader Limit* (*GLL*). On the other hand, different counters are also initialized in this step which are *Local Leader Limit Count* and *Global Leader Limit Count*.

2) Update of Individual SMs

In this step, each SM updates or changes its solution based on its current involvement, LL involvement and the involvement of randomly selected another member of that group. Eqs. (6) – (8) are used to update a SM if its selected under probability of *pr*.

$$SS_i = U(0,1)*(LL_k - SM_i) \otimes U(0,1)*(RSM_r - SM_i) \quad (6)$$

$$BSS_i = Cal\_BasicSS(SS_i) \quad (7)$$

$$SMnew_i = SM_i + BSS_i \quad (8)$$

In Eq. (6), LL_k is the Local Leader in kth group, SM_i is the i_{th} SM, RSM_r is another randomly selected SM within the group (i.e., $r \neq i$). At first, it calculates the SS between LL_k and SM_i; and select a portion using random *U (0,1)* for implication (say SS_1). Similarly, it calculates SS between RSM_r and SM_i; and select a portion for implication (say SS_2). *U (0,1)* is a random number uniformly distributed over [0,1]. Then those two SSs are merged into one SS; i.e., $SS_i = SS_1 \otimes SS_2$. Eq. (7) is used to optimize SS_i through *Cal_BasicSS* function; the outcome is the basic swap sequence BSS_i. The BSS_i is applied to SM_i using Eq. (8) with PS manner explained in the previous section and get new solution $SMnew_i$. The SM_i is updated with $SMnew_i$ if the new one is found better than the existing one. This procedure will continue for all the members of the k_{th} group.

After update individual SMs considering interaction within group they are again interacting with GL and another randomly selected member of the group. In this case, an SM takes the decision based on the value of probability if the SM will update or not. It ensures that the SM will partake a great opportunity to make itself better than the present solution. The probability for *i*th SM is calculated according to Eq. (9).

$$prob(i) = 0.9 * \frac{min\_cost}{cost(i)} + 0.1 \quad (9)$$

Here *cost(i)* is the tour cost of *i*th SM and *min_cost* is the minimum tour cost by the best SM of that group. The equation evaluates the probability by dividing the minimum cost found so far with the cost of each SM to generate a ratio below 1. The SMs are updated in this step using Eqs. (10) – (12).

$$SS_i = U(0,1)*(GL - SM_i) \otimes U(0,1)*(RSM_r - SM_i) \quad (10)$$

$$BSS_i = Cal\_BasicSS(SS_i) \quad (11)$$

$$SMnew_i = SM_i + BSS_i \quad (12)$$

Here, Eq. (10) calculates the SS between *GL* and SM_i, and SS of randomly selected RSM_r and SM_i. Eq. (11) is used to optimize SS_i that yields BSS_i; Finally, BSS_i is applied to SM_i using Eq.

(12) similar to Eq. (8); and $SMnew_i$ is obtained. If the tour cost is less than the older one, then the current solution SM_i is replaced with the new solution $SMnew_i$. Otherwise, the current solution is retained.

The way of individual SMs update in two different steps considering LL and GL, respectively, are the main steps in the proposed DSMO method. However, steps have similar operations. Eqs. (6)-(8) considered LL of the same group and a randomly selected solution while Eqs. (10)-(12) considered GL and a randomly selected solution. The procedure of updating an SM based on the LL is demonstrated in Fig. 1 for better understanding. In the figure, C_1, C_2, …, C_x are the index of a tour in solutions of operating spider monkey (*SM*), Local Leader (*LL*) and randomly selected another spider monkey (*RSM*). The SS generation using the experience of *SM* and *LL* is denoted by $SS_{LL} = (LL - SM)$; similarly, SS generation using *SM* and *RSM* is denoted by $SS_{RSM} = (RSM - SM)$. Using *U(0,1)*, a random portion of SS_{LL} is selected, i.e., $SS_{SLL} = \{U(0,1) * SS_{LL}\}$. A portion is also selected from SS_{RSM}, i.e., $SS_{SRSM} = \{U(0,1) * SS_{RSM}\}$. SS_{SLL} and SS_{SRSM} are merged together to $SS_M = SS_{SLL} \otimes SS_{SRSM}$. Then *Cal_BasicSS*() is applied to find BSS_M. The blue circle and the green circle indicate the SOs

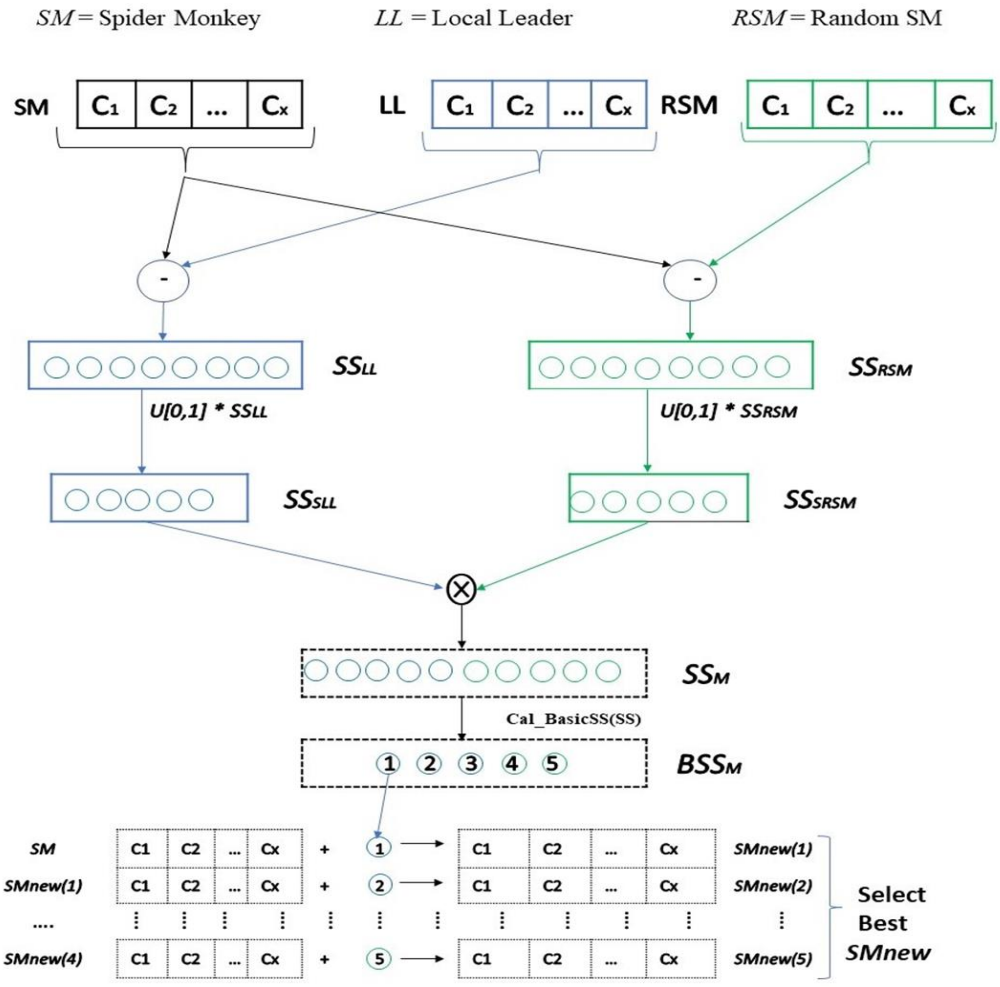

Figure 1. Demonstration of updating a Spider Monkey (*SM*) interacting with Local Leader (*LL*) and Random Spider Monkey (*RSM*).

generated by the *LL* and *RSM*, respectively. Lastly, apply SOs of BSS_M into the solution one after another and select the best solution (SM_{new}) among the tours from different for successive implications of SOs. The demonstration is shown in Fig. 1 is also applicable to update procedure of *SM* based on Global Leader considering *GL* (i.e., global best solution) instead of *LL*.

3) Update of LLs and GL

The Local Leaders (*LLs*) and Global Leader (*GL*) are updated in this step. For each group, the best SM having minimum tour cost is identified and is considered as the new Local Leader if it is better than the existing group leader; otherwise, the Local Leader is unchanged. If the Local Leader is not modified, the value of *Local Leader Limit Count* is increased by 1.

After the update of Local Leaders, the best LL (*bLL*) is identified and compared with the *GL*; update *GL* if *bLL* is better than the existing *GL*; otherwise, the Global Leader remains unchanged. Furthermore, *Global Leader Limit Count* is increased by 1 if the cost of *GL* is not modified.

4) Decision Phase of LL and GL

In the fourth step, if the *LL* is not updated for a certain period of time, Local Leader initializes all the group's members again and the *GL* takes the decision to divide the group into subgroups or join all the groups into one group.

Local Leader is supposed to update in each iteration to find a better solution. If it is not updated for *Local Leader Limit* times, all SMs of that group are updated either by randomly generated tour or interacting with the *GL* and *LL* using Eq. (13).

$$SMnew_i = SM_i + U(0,1)*(GL - SM_i) + U(0,1) * (SM_i - LL_k) \quad (13)$$

Here, *GL* is the Global Leader and LL_k is the Local Leader of the *kth* group. Following the Eq. (13), SM_i moves forward to the Global Leader and detaches from the Local Leader.

After checking all the Local Leaders limit counter, the Global Leader limit counter is checked. If the Global Leader fails to update for *Global Leader Limit* times, the Global Leader forms new groups from the existing groups by successively dividing them until *MG* is reached. Each time *GL* divides the group, a new *LL* is selected. The Global Leader can choose to combine the total population into a single group if *MG* groups are already formed and no more groups are allowed.

5) Termination and Outcome

At the end of each iteration, termination or stopping criteria of the DSMO is checked. The stopping criteria can be the number of iterations, fitness threshold etc. If the termination criterion is met, the TSP solution of the Global Leader is considered as the outcome of the proposed DSMO approach.

3.4 DSMO Algorithm

The proposed DSMO algorithm for solving TSP is shown in Algorithm 2. Detailed description of the algorithm is not given as the step wise operations are described already. The notations and inputs of the proposed algorithm are listed at the beginning. Like other population-based algorithms, it initializes the TSP solutions randomly. At first, it selects the Local Leader LL_k of every kth group and GL (Step 1). The main operations are performed in Step 2. At each iteration step, solutions monkeys are updated based on Local Leader and Global Leader in deferent steps. Algorithm 2.1 is used to update individual monkeys' solution SM_i based on Local Leader (LL_k) and Global Leader (GL). Local Leaders and Global Leader will

Algorithm 2: DSMO for Solving TSP

Input: List of Cities and their cost matrix

I	-	Total Number of Iterations
MG	-	Allowed Maximum Group
pr	-	Perturbation Rate
LLL	-	Local Leader Limit
GLL	-	Global Leader Limit
N	-	Total Number of Spider Monkeys

Output: An optimal solution of TSP

Variables:

t	-	Iteration Counter
g	-	Current Number of Groups
SM	-	Population of Spider Monkey
SM_i	-	ith Spider Monkey
LL_k	-	Local Leader of kth Group
LL_{new}	-	Updated Local Leader
LL	-	List of Local Leaders
GL	-	Global Leader
SS	-	Swap Sequence of Spider Monkey
BSS	-	Basic Swap Sequence of Spider Monkey
SM_{new}	-	Updated Spider Monkey
$prob(i)$	-	Probability of ith Spider Monkey
LLL_k	-	Local Leader Limit Counter of kth Group
bLL	-	best Local Leader
GLL_c	-	Global Leader Limit Counter

Step 1: Initialization

1. $t \leftarrow 1$
2. create N spider moneys and append them to SM
3. Assign each SM_i in SM with a random solution
4. $g \leftarrow 1$ // Initially Consider all SM into one group
5. Select Local Leader and Global Leader // *Both leaders are same due to single group*

Step 2: Main Operation

1. **while** $t \neq I$ **do**
2. Call **Algorithm 2.1** //Update of Spider Monkeys
3. Call **Algorithm 2.2** //Update of Local Leaders and Global Leader
4. Call **Algorithm 2.3** //Decision Phase of Local Leader and Global Leader
5. $t \leftarrow t + 1$

end while

Step 3: Return GL **as the result**

be updated if the newly generated solution is better than the current solution using Algorithm 2.2. Algorithm 2.3 is for decision phase of Local Leader and Global Leader where LL_k will initialize all the SM_i if the LL_k is not updated for a predetermined period of time and GL will split the population into some small sets or combine all the groups into one single set if the GL is not updating over a specific period of time. The iteration will continue until the termination

Algorithm 2.1: Update of Spider Monkeys based on Local Leader and Global Leader
Input: *SM, SM_i, LL_k,, LL, GL*

for each k^{th} *group* **do**
  for all SM_i in k^{th} group **do**
    If *U (0,1) >= pr* ***then***
      Calculate ***SS*** using Eq. (6)
      Calculate ***BSS*** using Eq. (7)
      Apply ***BSS*** into SM_i to calculate SM_{new} using Eq. (8)
      If *fitness (SM_{new}) > fitness (SM_i) then*
        $SM_i \leftarrow SM_{new}$
      end if
    end if
  end for
end for
for each k^{th} *group* **do**
  for all SM_i in k^{th} group **do**
    Calculate *prob(i)* using Eq. (9)
    If *U (0,1) <= prob(i)* ***then***
      Calculate ***SS*** using Eq. (10)
      Calculate ***BSS*** using Eq. (11)
      Apply ***BSS*** into SM_i to calculate SM_{new} using Eq. (12)
      If *fitness (SM_{new}) > fitness (SM_i) then*
        $SM_i \leftarrow SM_{new}$
      end if
    end if
  end for
end for

Algorithm 2.2: Update Phase of Local Leader and Global Leader
Input: *SM, SM_i, LL_k,, LL, GL, LLL, GLL*

for each k^{th} *group* **do**
  for all SM_i in k^{th} group **do**
    Calculate the fitness of SM_i
  end for
  $LL_{new} \leftarrow$ maximum (SM_i) *//select the highest fitness among all the spider monkeys in kth group*
  If *fitness (LL_{new}) > fitness (LL_k)* ***then***
    $LL_k \leftarrow LL_{new}$
    $LLL_k \leftarrow 0$
  else
    $LLL_k \leftarrow LLL_k + 1$
  end if
  $LL \leftarrow LL_k$
end for
bLL $\leftarrow$ maximum (*LL*) *//select the best Local Leaders heaving highest fitness among all Local Leaders*
If *fitness (bLL) > fitness (GL)* ***then***
  $GL \leftarrow bLL$
  $GLL_c \leftarrow 0$
else
  $GLL_c \leftarrow GLL_c + 1$
end if

criterion (i.e., total iterations) is met. Finally, TSP tour in *GL* is considered as the outcome of DSMO (Step 3).

Algorithm 2.3: Decision Phase of Local Leader and Global Leader
Input: g, SM_i, LL_k,, LLL, GLL, LLL_k, GLL_k
1. **for each** k^{th} *group* **do**
2. **If** $LLL_k > LLL$
3. $LLL_k \leftarrow 0$
4. **for all** SM_i in k^{th} group **do**
5. **If** *U (0,1) >= pr* ***then***
6. Initialize SM_i randomly
7. **else**
8. Initialize SM_i using **Eq. (13)**
9. **end if**
10. **end for**
11. **end if**
12. **end for**
13. **If** $GLL_c > GLL$
14. $GLL_c \leftarrow 0$
15. **If** $g < MG$ **then**
16. Divide the spider monkeys into $g + 1$ number of groups.
17. **else**
18. Disband all the groups and Form a single group.
19. $g \leftarrow 1$
20. **end if**
21. **end if**

3.5 Significance of Proposed DSMO

DSMO follows self-organization as well as division of labour properties for finding clever operations for TSP. Proposed DSMO introduces SO and SS based interactions to update individual TSP tours represent by spider monkeys. In the method, a spider monkey updates its solution using subgroup leader (called Local Leader), the main group leader (called Global Leader), and self-experience. When the Global Leader divides the groups into subgroups, it implies the divisions of labour property. Due to multiple sources consideration of SOs calculation, diverse solutions are brought in at different iterations and hence easy to find a better solution of a TSP problem.

DSMO has a distinct structure quite different from the existing SI methods including ACO and Velocity Tentative PSO (VTPSO). DSMO updates an SM in two different stages. In the first stage, each and every SM is updated interacting with Local Leader and randomly selected another SM following Eqs. (6)-(8). In the second stage, each and every SM is updated interacting with the Global Leader and randomly selected another SM following Eqs. (10)-(12). On the other hand, VTPSO as well as PSO update a particle in a single stage interacting with the global best and personal best solution [90], [92]. In contrast, solutions are updated based on the pheromone on the paths in ACO that is different from other SI methods. Therefore, multiple interaction stages might be beneficial to improve solutions with respect to other existing methods.

4. EXPERIMENTAL STUDIES

The effectiveness of the proposed DSMO is assessed by applying to benchmark TSPs. The outcomes of DSMO are also compared with the well-known metaheuristic algorithms applied to TSP which are ACO[87], VTPSO[33] and Artificial Bee Colony with SS (ABCSS)[32]. ACO is one of the first SI algorithms applied to TSP and had been well-studied for TSP [87]. In ACO, each ant maintains a tabu list of unvisited cities and calculates transition probability of cities individually; moves to one of the cities; and the process continue until completion of the tour visiting all the cities. Pheromone update is also computational procedure at the end of each iteration after all the ants completed the tours. Each element (e.g., particle, bee or monkey) in VTPSO, ABCSS and the proposed DSMO, always contains a complete feasible solution and interacts with others elements following distinctive operations to improve the solution. The common property of the three methods is the use of Swap Sequence (SS) in their operations. VTPSO is the PSO based method for TSP which transformed PSO operations for TSP with SS including Partial Search (PS), single node adjusted and block node adjusted [90]. ABCSS is the most recent TSP algorithm with Artificial Bee Colony with SS operation[25]. It uses SS based rules in employed bee and onlooker bee phases, and uses K-opt operation in scout bee phase and final outcome stage. On the other hand, proposed DSMO considered SS based PS related operations like VTPSO in its different steps considering Local Leader and Global Leader.

Following subsections provide description on the benchmark instances and experimental settings; experimental analysis on selected instances, and finally, experimental results of the proposed method on benchmark instances comparing with other methods.

4.1 Benchmark TSP Data and Experimental Setup

As a test bench, a suite of 45 benchmark TSP instances are chosen from TSPLIB [94], the well-known benchmark repository for TSPs. The size of the selected instances varies from 14 to 493 cities which makes them diverse test beds. A numeric value with an instance indicates the number of cities in the instance. Like st70 and rat99 have 70 and 99 cities, respectively. In the dataset, coordinates of individual cities are given for a particular instance which need to process to use any system. The cost matrix is prepared using coordinates of cities which is then used to calculate the tour cost. A tour cost of a TSP solution is the sum of the cost distances of individual links considered in the tour. Tour cost is considered as the fitness value in TSP by any optimization method where the minimum of the tour cost is assumed as good.

The parameters of DSMO are maximum number of groups (*MG*); *Local Leader Limit* (*LLL)* and *Global Leader Limit* (*GLL);* and perturbation rate (*pr).* In the experiment, *MG* is set to 5; both *LLL* and *GLL* are varied 50 to 100; *pr* is initialized with 0.1. In ACO, parameter values for alpha and beta are 1 and 3, respectively. The parameters of ABCSS are considered as suggested in the paper or found better than those; upper limit of a food source counter and maximum trail of K-opt were set to 5 and 10, respectively. These parameters values are not optimal though but are chosen for simplicity enabling fairness in comparison.

The methods ACO, VTPSO and proposed DSMO are implemented on Visual C++ of Visual Studio 2017 on Windows 10 OS environment. Tests have been led on a desktop computer (HP ProDesk 490 G2 MT, Intel(R) Core (TM) i7-4790M CPU @ 3.60 GHz, RAM 8 GB).

4.2 Experimental Analyses

Proposed DSMO is a novel population-based SI method; therefore, like other SI methods, population size *M* (i.e. the number of monkeys in population) and total iteration are the two most important parameters towards good outcome. In this section, the effects of population size and maximum iteration on DSMO are investigated on three selected instances which are berlin52, rat99 and gr137. The selected instances are different in size and are studied many existing studies. ACO and VTPSO are also included in the analyses for better understanding of DSMO compared to the algorithms.

1) Performance Improvement and Time Requirement with Iteration

This section represents the comparative performance analysis and required training time among ACO, VTPSO, ABCSS, and DSMO while training up to a certain iteration. For a fair comparison, the population sized was fixed at 100 for all four methods. Fig. 2 demonstrates the tour costs of ACO, VTPSO, ABCSS and DSMO for sample runs up to 150 iteration on berlin52, rat99 and gr137 instances.

It is observed from Fig. 2 that all the method showed large tour costs for any instance at the beginning of the training (i.e., in first iteration) and improved with iteration. However, ABCSS show very high tour cost with respect to ACO, VTPSO and DSMO; and performance gradually improved with iteration. As an example, for berlin52 after first iteration, tour cost of ABCSS is 23062.41; whereas, the tour costs at that point for ACO, VTPSO and DSMO are 8207.61, 12197.49 and 10133.75, respectively. For the same instance, the tours cost for ACO, VTPSO, ABCSS and VTPSO are 8093.35, 7977.08. 8222.05 and 8238.60, respectively. The shape of achieved tour costs curves by the methods for rat99 and gr137 instances are also similar. It is notable for all three instances that ACO and VTPSO showed invariant performance after several iterations (e.g., after 50 iteration for berlin52) whereas ABCSS and DSMO is shown to improve with respect iteration up to 150 iteration. This indicates first convergence of ACO and VTPSO but better outcomes by ABCSS and DSMO with large number of iterations. The scenario replies the effectiveness of employed heuristics used in ABCSS and DSMO.

The proposed DMSO and other investigated methods (i.e., ACO, VTPSO and ABCSS) are population-based bio-inspired algorithms where the computational time of a method depends on the iterations as well as the population size. The computational complexity of a particular method depends on its deployed operators. The time requirement over the iterations for ACO, VTPSO, ABCSS and DSMO are measured for the results presented in Fig. 2 and are found similar for all the three instances that increases linearly with iterations but have different rates for different algorithms.

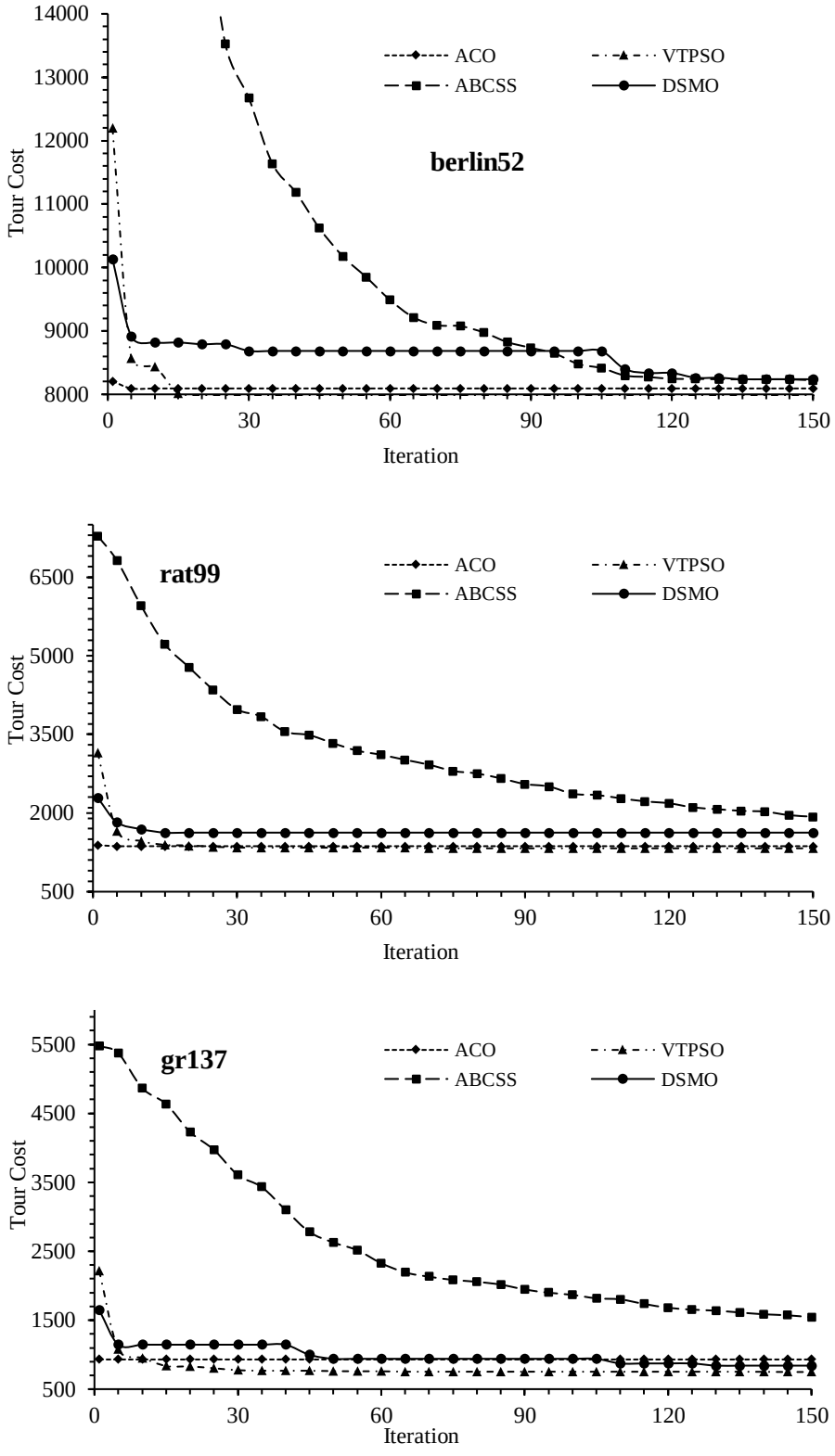

Figure 2. Tour cost improvement with iteration.

Figure 3 demonstrates the time requirement with iterations for gr137 instance as a sample case. According to the figure, ACO is computationally extensive and ABCSS is the most efficient. In ACO, calculation of transition probability for unvisited cities by each ant and

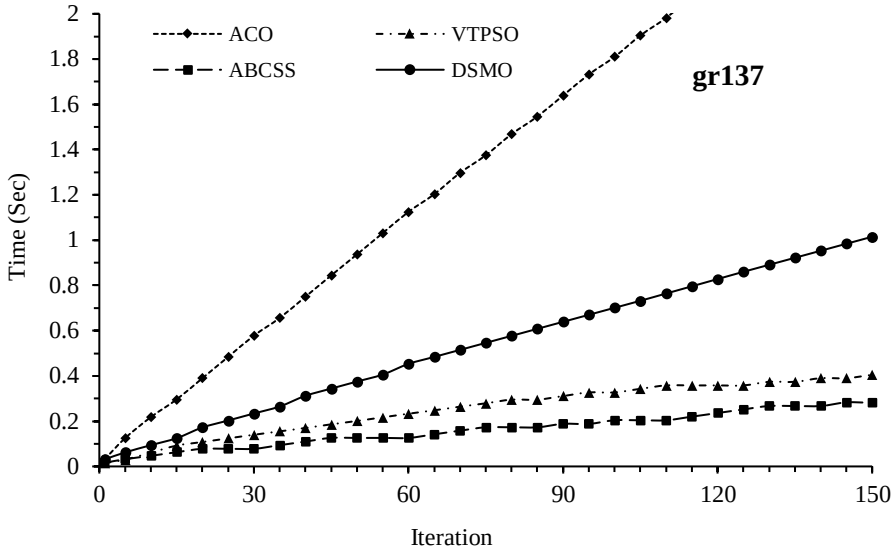

Figure 3. Required time with iteration.

recalculation again after moving to a city until tour completion incur large computational cost. Pheromone update is also part of computation cost; therefore, computational time of ACO is high. SS based velocity operation and node adjustment are the main computational burden in VTPSO. Proposed DSMO performed VTPSO like operations on each element in two different steps considering Local Leader and Global Leader and have computation burden of other tasks such as subgrouping; hence, computational time of DSMO is larger than VTPSO. ABCSS used K-opt heuristic based operation (in scout bee phase and final stage) in addition to SS based different rules deployment in employed bee and onlooker bee phases; the operations of ABCSS seem efficient compared to the operations of DSMO.

2) Impact of Population Size Variation

The proposed DMSO and other considered methods (i.e., ACO, VTPSO and ABCSS) are population-based bio-inspired algorithms and this section compares the efficiency of the methods for varying population size. The population size considered for ACO, VTPSO and ABCSS from 10 to 500. In contrary, the population size started in DSMO from 20 to manage *MG* as 5. For fair comparison, fixed 500 iterations were considered for all three methods.

Figure 4 presents the tour cost for different population size on berlin52, rat99 and gr137 instances. The presented results are the average of 10 individual runs. Standard Deviation (SD) values of individual results are shown as vertical line bars on the average value. It is noticeable from the figure that ABCSS performed very badly at small population size (e.g., 10, 20) and improved with increasing the population size. For rat99 and gr137, which are relatively large instances, ABCSS showed worst outcomes for low population size and showed competitive performance with VTPSO and DSMO for population size 100 and onward. Meanwhile, ACO is shown an almost unchanged performance for a specific instance. In ACO, each ant starts from a city and initially pheromone values are equal in all the path; therefore, ants larger than

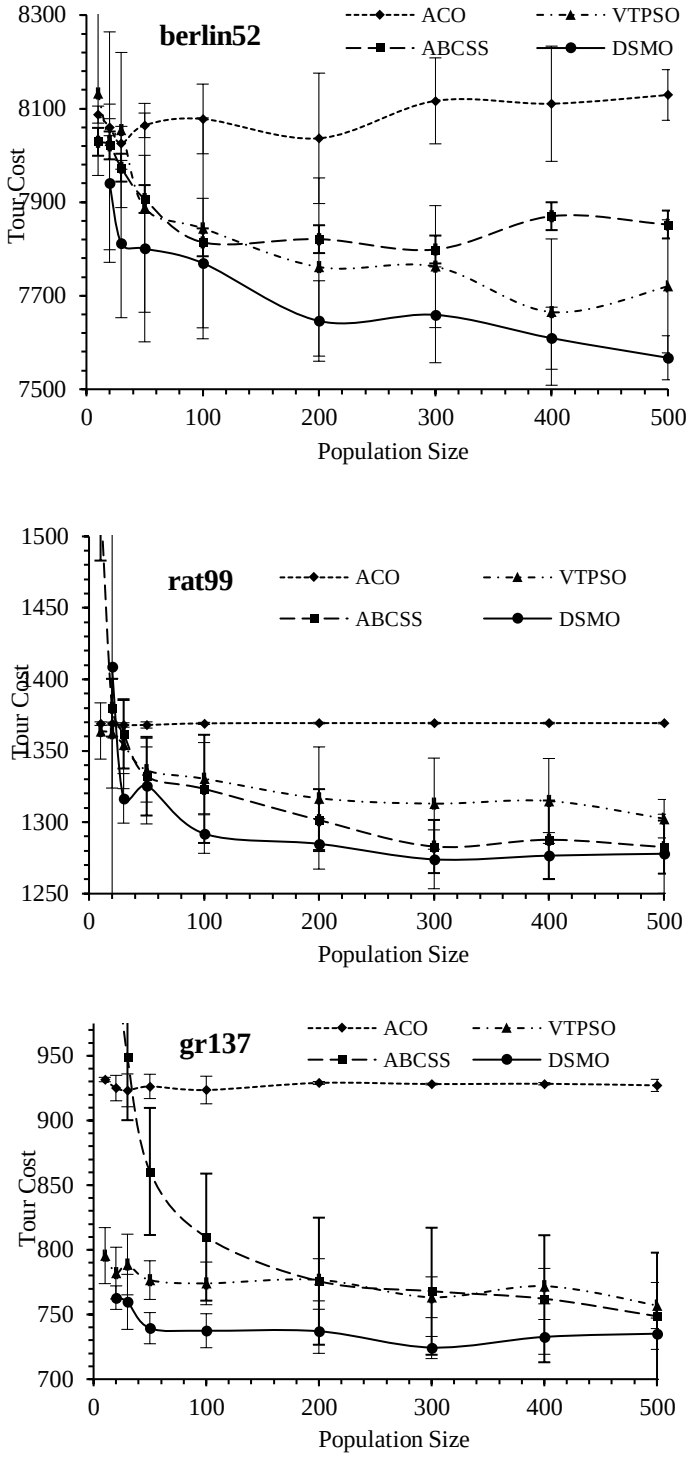

Figure 4. Impact of population size on tour cost.

total city contains several duplicate paths and failed to improve solution at all. On the other hand, due to pheromone-based heuristic, ACO is shown more contestant results over different runs; therefore, SD values of ACO are comparatively very low with respect to the values of VTPSO, ABCSS and DSMO.

It looks interesting from the figure that DSMO is the best in general for any population size for all three instances. As an example, for the gr137 instance ACO, VTPSO, ABCSS and DSMO showed tout costs of 925.05, 782.03, 1034.01 and 763.03, respectively when the population size was 20. The tour costs became 923.57, 774.07, 809.83 and 737.44 for population size increased to 100. VTPSO and ABCSS achieved the best tour costs of 756.96 and 748.69, respectively at population size of 500. On the other hand, the achieved best tour cost of DSMO is better than VTPSO / ABCSS; the achieved value is 724.46 at population size 300. Comparing the results of all the instances, proposed DSMO has shown better results in different population sizes outperforming other the methods.

4.3 Experimental Results and Performance Comparison

The experimental results of the proposed DSMO in solving the suite of 45 benchmark TSPs are compared with ACO, VTPSO and ABCSS. For the fairness in comparison, the population size was kept between 100 and 300 since the outcomes did not change significantly for larger population size. On the other hand, the iteration was fixed at 500 for ACO, VTPSO and DSMO as termination criteria; for ABCSS the iteration was varied up to 1000. The best outcome for different population sizes (i.e., 100, 200 and 300) for a particular instance by a method is considered in the comparison.

Table I compares the performance of ACO, VTPSO, ABCSS and DSMO on 20 different runs for each TSP instance. For a better evaluation, SD values of 20 runs are placed with average value for a particular instance. Since the best solution having the minimum tour cost from different runs may use as the outcome, minimum tour cost from different runs are also included in performance evaluation and placed in the table. For both average and minimum tour costs, the best value (i.e., smallest cost) among the three methods was shown in bold-face type and the worst value (i.e., biggest cost) was indicated by underlined face type. Average on all the instances and best/worst count are shown at the bottom of the table. It indicates how many instances a method gave the best/worst result. A pairwise summary Victory-Draw-Defeat from Table I for all the methods are also presented in Table II for the pairwise algorithm comparison.

On the basis of average tour cost in Table I for all the instances, DSMO is the best and ABCSS is the worst among the four methods. The average tour cost of DSMO was 24342.29 for all 45 instances. The accomplished average tour costs of ACO, VTPSO and ABCSS were 26310.95, 24682.71 and 29452.21, respectively. In the Best/Worst count for individual instances, DSMO showed the best results for 20 cases out of 45 instances; and remaining 25 cases it showed competitive results and did not showed the worst for any one. ABCSS showed the best average tour costs for 17 cases outperforming ACO (best for no one) and VTPSO (the best for only two cases). It is observable that for which ABCSS showed best results are small in size (and less than 150) and other small sized instances ABCSS outperformed or competitive to other methods. ACO, on the other hand, is found worst for small sized instance except gr17; all four methods showed the same tour cost of 2332.58 for the instance. In general, VTPSO,

TABLE I. PERFORMANCE COMPARISON AMONG ACO, VTPSO AND DSMO TO SOLVE BENCHMARK TSPS.

SL	TSP Instance	Average Tour Cost (Standard Deviation)				Minimum Tour Cost			
		ACO	VTPSO	ABCSS	DSMO	ACO	VTPSO	ABCSS	DSMO
1	burma14	31.21 (0)	30.87 (0)	30.87 (0)	30.87 (0)	31.21	30.87	30.87	30.87
2	ulysses16	77.13 (0)	73.99 (0)	74 (0)	73.99 (0)	77.13	73.99	73.99	73.99
3	gr17	2332.58 (0)	2332.58 (0)	2332.58 (0)	2332.58 (0)	2332.58	2332.58	2332.58	2332.58
4	gr21	2953.46 (2.67)	2672.27 (0)	2672.27 (0)	2672.27 (0)	2949.81	2672.27	2672.27	2672.27
5	ulysses22	86.71 (0.45)	**75.33** (0.06)	75.37 (0.18)	75.4 (0.02)	84.78	75.31	75.31	75.31
6	gr24	1267.13 (0)	1249.78 (0)	1260.81 (17.48)	1249.78 (0)	1267.13	1249.82	1249.82	1249.82
7	fri26	646.48 (0)	635.58 (0)	635.58 (0)	635.58 (0)	646.48	635.58	635.58	635.58
8	bayg29	9964.78 (0)	9078 (17.61)	9107.63 (43.25)	**9074.15** (0)	9964.78	9074.15	9074.15	9074.15
9	hk48	12733 (78.99)	11376 (281.4)	**11283.3** (144.19)	11296 (215.76)	12699.86	11104.67	11104.67	11104.67
10	eil51	504.97 (1.86)	439.87 (5.65)	437.01 (4.98)	**436.96** (4.73)	499.92	429.51	428.98	**428.86**
11	berlin52	8078.82 (49.05)	7728 (161.8)	7807.86 (177.55)	**7633.6** (85.4)	7870.45	7544.37	7544.37	7544.37
12	st70	749.19 (8.28)	711.5 (14.57)	**690.5** (5.35)	702.64 (15.04)	734.19	682.57	682.57	677.11
13	eil76	598.99 (8.22)	569.1 (9.12)	**561.48** (7.21)	572.7 (7.56)	581.42	559.25	**550.24**	558.68
14	pr76	127181.5 (172.01)	112549.2 (1420)	**109758.57** (850.64)	111299.3 (2050.48)	127025.9	109586.1	108879.7	**108159.4**
15	gr96	588.92 (7.93)	537.92 (11.89)	**523.31** (8.93)	530.45 (7.29)	563.77	515.27	**512.2**	518.38
16	rat99	1369.2 (0.85)	1325.8 (34.83)	**1265.93** (13.54)	1291.93 (21.07)	1366.3	1256.25	1242.32	**1225.56**
17	kroa100	24659.09 (66.87)	22400.48 (529.13)	**21878.83** (455.63)	22024.27 (508.89)	24504.9	21307.44	21299	**21298.21**
18	kroB100	24937 (284.08)	23236.43 (522.63)	**22707.96** (259.83)	23022.37 (277.32)	24664.13	22475.67	**22229.71**	22308
19	rd100	9406.56 (91.14)	8502.73 (167.7)	**8207.8** (172.52)	8377.76 (209.4)	9120.92	8094.75	**7944.32**	8041.3
20	eil101	732.86 (7.48)	679.14 (13.29)	**662.63** (7.13)	674.4 (10.97)	715.35	653.16	**646.05**	648.66
21	lin105	15785.33 (421.1	15684.64 (610.38)	**14766.55** (263.01)	15114 (500.76)	15364.58	14581.58	14406.12	**14383**
22	pr107	46535 (129.95)	45287.9 (1207.52)	**44927.27** (319.03)	45666.99 (1300.43)	46317.71	44436.25	44525.68	**44385.86**
23	pr124	65145.25 (0)	63939.97 (1739.4)	**59772.68** (516.56)	62443.49 (1644.93)	65145.25	61076.73	**59030.74**	60285.21
24	pr136	110946.3 (322.83)	102945.7 (1688.2)	**101795.57** (1916.45)	102872 (2855.28)	110872.2	99247.01	97853.91	**97538.68**
25	gr137	927.44 (9.99)	765.21 (22.25)	738.5 (14.44)	**736.67** (15.61)	896.07	714.18	713.91	**709.48**
26	kroA150	31170.42 (365.45)	28262.98 (455.95)	**27971.36** (554.5)	28354.09 (524.91)	30546.06	27232.1	**26981.98**	27591.44
27	kroB150	29922.94 (961.65)	27987.85 (620.72)	27653.49 (509.24)	**27576.16** (625.26)	29124.59	**26579.73**	26760.79	26601.94
28	pr152	79423 (198.43)	76272.37 (1199.9)	**76097.48** (904.45)	76526.77 (1663.08)	79153.02	74414.17	74337.62	**74243.91**
29	u159	47615.78 (242.53)	45441.55 (1178.06)	**45234.92** (1212.54)	42598.3 (0)	47514.43	43579.82	42862.51	**42598.3**
30	rat195	2555.3 (26.41)	2554.1 (47.56)	2579.27 (44.14)	**2488.55** (50.48)	2534.83	2452.92	2469.31	**2372.89**
31	d198	17374 (111.15)	16489.28 (206.79)	16483.73 (162.08)	**16270.47** (171.2)	17301.47	16066.44	16270.22	**15978.13**
32	kroA200	34547.69 (0)	**31502.33** (531.41)	31938.81 (656.84)	31828.64 (652.32)	34547.69	30602.81	30701.86	**30481.35**
33	kroB200	35109.8 (280.51)	31923.15 (553.02)	32208.73 (526.77)	**31781.62** (487.39)	34207.79	30767.52	**31508.85**	30716.5
34	gr202	554.58 (6.19)	513.04 (7.24)	520.13 (6.11)	**508.81** (4.08)	545.33	**497.02**	507.27	501.83
35	tsp225	4544.82 (54.17)	4210.65 (66.44)	4276.92 (72.8)	**4162.79** (66.08)	4396.39	4095.01	4140.24	**4013.68**
36	pr226	90501.47 (0.02)	88031.31 (3461.96)	87400.6 (3482.64)	**85935.69** (2105.13)	90501.46	**81050.23**	82266	83587.98
37	gr229	1915.45 (21.82)	1736.35 (26.19)	1764.79 (22.28)	**1730.46** (20.05)	1865.63	**1676.46**	1713.54	1683.45
38	gil262	2787.01 (31.83)	2631.69 (39.44)	2839.11 (73.01)	**2627.87** (42.39)	2730.52	2547.16	2713.75	**2543.15**
39	pr299	57509.4 (444.01)	53154.34 (1389.36)	67620.95 (2092.63)	**51747.99** (863.32)	56700.65	**50571.83**	64464.76	50579.82
40	lin318	48583.74 (560.27)	46209.53 (773.01)	61902.84 (3824.65)	**45460.25** (660.47)	47442.95	44724.38	55744.52	**44118.66**
41	linhp318	48333.33 (415.2)	46329.87 (948.05)	60853.66 (2306.85)	**45730.57** (929.73)	47577.77	44337.02	56834.19	**43831.44**
42	fl417	13618.22 (209.51)	12980.24 (311.88)	27237.13 (2479.9)	**12950.77** (360.99)	13296.85	12376.53	22253.99	**12218.98**
43	gr431	2374.72 (16.03)	2061.72 (30.34)	3630.03 (201.42)	**2042.77** (24.08)	2334.63	2021.95	3284.99	**1993.15**
44	pr439	127523 (224.55)	119442.2 (2770.58)	244192.44 (21576.5)	**116379.2** (2462.82)	127228	**112088**	206233.14	112105.2
45	d493	39789 (407.83)	38159.18 (603.84)	78968.31 (5776.85)	**37861.14** (426.97)	39254	37132.09	68556.68	**36844.63**
Average		**26310.95**	**24682.71**	**29452.21**	**24342.29**	**26113.35**	**23671.61**	**27474.34**	**23568.14**
Best/Worst		**0/27**	**2/0**	**17/16**	**20/0**	**0/28**	**6/0**	**8/11**	**20/0**

ABCSS and DSMO showed competitive performance for small sized instances; the methods showed the same performance (e.g., burma14, gr21, fri26) for several small sized instances.

But ABCSS found inferior to others for large sized instances; rat195 and larger cases (i.e., 16 largest instances), it showed worst tour costs. On the other hand, DSMO showed significantly better performance for large sized instances; it showed the best tour costs for all the instances having instance size larger than 200 (i.e., gr202 to d493).

SD value with average achieved tour cost for a particular in Table I reflects the variability of 20 individual runs by a method in solving an instance. For solving TSP, ACO starts placing different ants in different cities, its initialization did not differ much among individual runs. Therefore, the tour costs of ACO in different runs were found consistent showing lower SD values with respect to other well-known existing methods including VTPSO [90]. For small instances (where the number of cities is small) like ulysses16, bays29 the tour cost for all 20 individual runs are the same, so the SD of the average tour cost was zero. On the other hand, VTPSO gives higher variant outcomes among different runs showing the larger SD value. But it is interesting to observe from the results presented in Table I that the SD value of DSMO is lower than VTPSO in general. On the other hand, SD values of ABCSS are shown lower values for small sized instances (for which the method performed well) but the method showed larger SD values for large sized instances (for which the performed worse). As an example, ACO achieved an average tour cost of 1369.2 with SD of 0.85 in case of rat99. For the same instance, VTPSO, ABCSS and DSMO achieved average tour costs of 1325.8 (with SD 34.83), 1265.93 (with SD 13.54) and 1291.93 (with SD 21.07), respectively.

The best result may use the outcome of a method after several runs; therefore, the achieved minimum tour cost from 20 runs were also compared in Table I. For small sized 12 instances, VTPSO, ABCSSS and DSMO showed competitive performance and all three methods showed same tour cost for nine instances whose size below 50 (i.e., burma14 to hk48). For rest 33 instances, best minimum tour cost achievements are distributed as 20, 8, and 6 cases for DSMO, ABCSS and VTPSO, respectively. ABCSS showed the best minimum tour costs for small sized instances, i.e., instance size less than 150; DSMO is generally showed the best minimum tour cost for large sized instances. Again, ACO showed worst minimum tour costs in 28 small sized instances showing best result for no one. On the other hand, ABCSS showed worst results for 11 large sized instances; but VTPSO and DSMO did not showed worst for none of the instance. The results revealed that DSMO is an effective method to TSP.

Table II presents pairwise Victory-Draw-Defeat summary among the methods based on the results presented in Table I and discussed already. On the basis of average tour costs presented in Table II(A), ACO was worse than DSMO and VTPSO in all the cases except gr17. For gr17, all the methods achieved same tour cost of 2332.58 as seen in Table I. ABCSS outperformed ACO for 35 cases but it was inferior to ACO nine cases. ABCSS outperformed VTPSO in 24 cases whereas it found inferior for other 19 cases. Between DSMO and VTPSO, DSMO was better than VTPSO for 33 cases and DSMO was inferior to VTPSO six cases only; and the rest six cases both the methods give the same result. DSMO also found better than ABCSS showing better for 24 cases; among rest 21 cases ABCSS outperformed for 18 cases and showed same result of DSMO for three cases. Victory-Draw-Defeat summary on the basis of average

TABLE II. PAIRWISE VICTORY-DRAW-DEFEAT SUMMARY OF RESULTS PRESENTED IN TABLE I

(A) SUMMARY ON AVERAGE TOUR COST

Method	ACO	VTPSO	ABCSS	DSMO
ACO	-	44-1-0	35-1-9	44-1-0
VTPSO		-	24-2-19	33-6-6
ABCSS			-	24-3-18

(B) SUMMARY ON MINIMUM TOUR COST

Method	ACO	VTPSO	ABCSS	DSMO
ACO	-	44-1-0	37-1-7	44-1-0
VTPSO		-	16-11-18	27-10-8
ABCSS			-	27-10-8

minimum tour costs presented in Table II(B) are found similar to Table II(A) and DMSO is outperformed individually ACO, VTPSO and ABCSS for 44, 27 and 27 cases, respectively.

Wilcoxon signed ranks test was carried out to decide the importance of variance in results of DSMO with ACO, VTPSO and ABCSS. It is a nonparametric procedure employed in hypothesis testing situations, involving a design with two samples to detect significant differences between two sample means, that is, the behaviour of two algorithms [95]. Table III shows the test outcomes between DSMO and the rest of algorithms. Equal values are discarded (EVD) from the test; and then R^+ (sum of positive ranks) and R^- (sum of negative ranks) are measured. Test value T is the minimum of positive and negative rank sums and its critical value (i.e., T_{Crit}.) is collected from the distribution table. Table III shows the R+, R-, and p-values computed for all the pairwise comparisons relating to DSMO. The values have been computed according to [95], [96]. As the table states, DSMO shows a significant improvement over ACO and VTPSO with a level of significance $\alpha = 0.001$, and over ABCSS with $\alpha = 0.2$. The competitive performance of ABCSS with DSMO is logical because it uses different SS based rules and K-opt operation in different phases. Finally, DSMO is undeniably a good choice to solve TSP taking into account of the results presented in Table I and significant test summarized in Table II and Table III.

TABLE III. WILCOXON SIGEND RANKS TEST OF DSMO W.R.T. ACO, VTPSO AND ABCSS ON BASIS OF AVERAGE TOUR COSTS PRESNETED IN TABLE I..

Comparison	No of EVD from test	R^+	R^-	$T = Min\ (R^+, R^-)$	$T_{Crit.}$ for (45- EVD)	p-value
DSMO versus ACO	1	990	0	0	220	7.6159E-09
DSMO versus VTPSO	6	676	104	104	161	6.57594E-05
DSMO versus ABCSS	3	560	343	343	195	0.174893237

5. CONCLUSIONS

TSP is one of the popular combinatorial problem and new techniques are being proposed to solve it. In this paper, a new method namely Discrete Spider Monkey Optimization (DSMO) is proposed to solve TSP. Standard SMO is a recently developed SI method for function optimization. DSMO is developed modifying basics of SMO as well as introducing new

operations to be applicable to TSP. In DSMO, individual spider monkey represents a TSP solution. In every iteration, the spider monkey tries to update considering the Local Leader and the Global Leader solutions. Swap Sequence (SS), Swap Operator (SO) and related procedures are developed and applied to improve TSP solutions of individual monkeys. Grouping spider monkeys and updating a monkey interacting with group leader (i.e., best within a group) and Global Leader (i.e., best among all the groups) in two different stages is the significant property of DSMO. At first, SS with several SOs is measured for a particular monkey and the best solution is considered for the implication of several or all SOs of the SS. DSMO has been tested on a large suite of TSPs from benchmark repository and revealed as the best suited method for solving TSP. Idea of solving TSP in DMSO based on the intelligent behaviour of spider monkeys has opened a new direction to solve other discrete optimization tasks. Solving different scheduling and routing problems extending conceiving ideas from DMSO might be interesting and remained as a future study.

REFERENCES

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, 1989.

[2] A. E. Eiben and J. Smith, "From evolutionary computation to the evolution of things," *Nature*, vol. 521, no. 7553, pp. 476–482, May 2015.

[3] R. M. Brady, "Optimization strategies gleaned from biological evolution," *Nature*, vol. 317, no. 6040, pp. 804–806, Oct. 1985.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for optimization from social insect behaviour," *Nature*, vol. 406, no. 6791, pp. 39–42, Jul. 2000.

[5] K. Luo, "Enhanced grey wolf optimizer with a model for dynamically estimating the location of the prey," *Appl. Soft Comput. J.*, vol. 77, pp. 225–235, 2019.

[6] F. B. Ozsoydan and A. Baykasoglu, "A swarm intelligence-based algorithm for the set-union knapsack problem," *Futur. Gener. Comput. Syst.*, vol. 93, pp. 560–569, Apr. 2019.

[7] M. S. Rahman Tanveer, M. J. Islam, and M. Akhand, "A Comparative Study on Prominent Swarm Intelligence Methods for Function Optimization," *Glob. J. Technol. Optim.*, vol. 07, no. 03, pp. 1–8, 2017.

[8] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, vol. 4, pp. 1942–1948.

[9] H.-C. Tsai and Y.-H. Lin, "Modification of the fish swarm algorithm with particle swarm optimization formulation and communication behavior," *Appl. Soft Comput.*, vol. 11, no. 8, pp. 5367–5374, Dec. 2011.

[10] F. Vandenbergh and A. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Inf. Sci. (Ny).*, vol. 176, no. 8, pp. 937–971, Apr. 2006.

[11] S. Imran Hossain, M. A. H. Akhand, M. I. R. Shuvo, N. Siddique, and H. Adeli, "Optimization of University Course Scheduling Problem using Particle Swarm Optimization with Selective Search," *Expert Syst. Appl.*, vol. 127, pp. 9–24, Aug. 2019.

[12] A. Alexandridis, E. Paizis, E. Chondrodima, and M. Stogiannos, "A particle swarm optimization approach in printed circuit board thermal design," *Integr. Comput. Aided. Eng.*, vol. 24, no. 2, pp. 143–155, Mar. 2017.

[13] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. Syst. Man Cybern. Part B*, vol. 26, no. 1, pp. 29–41, 1996.

[14] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[15] S. Sharma and P. Bhambu, "Artificial Bee Colony Algorithm: A Survey," *Int. J. Comput. Appl.*, vol. 149, no. 4, pp. 11–19, Sep. 2016.

[16] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108–132, Aug. 2009.

[17] J. Simpson and K. von Frisch, "The Dance Language and Orientation of Bees," *J. Anim. Ecol.*, vol. 38, no. 2, p. 460, Jun. 1969.

[18] X.-S. Yang, "Firefly Algorithms," in *Nature-Inspired Optimization Algorithms*, Elsevier, 2014, pp. 111–127.

[19] K. N. Krishnanand and D. Ghose, "Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions," *Swarm Intell.*, vol. 3, no. 2, pp. 87–124, Jun. 2009.

[20] X. S. Yang, "Bat algorithm for multi-objective optimisation," *Int. J. Bio-Inspired Comput.*, vol. 3, no. 5, p. 267, 2011.

[21] F. B. Ozsoydan and A. Baykasoglu, "Analysing the effects of various switching probability characteristics in flower pollination algorithm for solving unconstrained function minimization problems," *Neural Comput. Appl.*, vol. 2, pp. 1–15, 2018.

[22] M. A. Al-Betar, M. A. Awadallah, I. Abu Doush, A. I. Hammouri, M. Mafarja, and Z. A. A. Alyasseri, "Island flower pollination algorithm for global optimization," *J. Supercomput.*, Mar. 2019.

[23] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Syst.*, vol. 22, no. 3, pp. 52–67, Jun. 2002.

[24] H. Chen, B. Niu, L. Ma, W. Su, and Y. Zhu, "Bacterial colony foraging optimization," *Neurocomputing*, vol. 137, pp. 268–284, Aug. 2014.

[25] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, 2017.

[26] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[27] S. He, Q. H. Wu, and J. R. Saunders, "Group Search Optimizer: An Optimization Algorithm Inspired by Animal Searching Behavior," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 973–990, Oct. 2009.

[28] J. C. Bansal, H. Sharma, S. S. Jadon, and M. Clerc, "Spider Monkey Optimization algorithm for numerical optimization," *Memetic Comput.*, vol. 6, no. 1, pp. 31–47, Mar. 2014.

[29] F. B. Ozsoydan, "Artificial search agents with cognitive intelligence for binary optimization problems," *Comput. Ind. Eng.*, vol. 136, pp. 18–30, Oct. 2019.

[30] R. Matai, S. Singh, and M. Lal, "Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches," in *Traveling Salesman Problem, Theory and Applications*, InTech, 2010, pp. 1–24.

[31] M. A. H. Akhand, S. Akter, S. Sazzadur Rahman, and M. M. Hafizur Rahman, "Particle Swarm Optimization with partial search to solve Traveling Salesman Problem," in *2012 International*

Conference on Computer and Communication Engineering (ICCCE), 2012, pp. 118–121.

[32] I. Khan and M. K. Maiti, "A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem," *Swarm Evol. Comput.*, vol. 44, no. November 2016, pp. 428–438, Feb. 2019.

[33] M. A. H. Akhand, S. Akter, M. A. Rashid, and S. B. Yaakob, "Velocity tentative PSO: An optimal velocity implementation based particle swarm optimization to solve traveling salesman problem," *IAENG Int. J. Comput. Sci.*, vol. 42, no. 3, pp. 1–12, 2015.

[34] C. R. Carpenter, "Behavior of Red Spider Monkeys in Panama," *J. Mammal.*, vol. 16, no. 3, p. 171, Aug. 1935.

[35] R. O. Deaner, C. P. van Schaik, and V. Johnson, "Do Some Taxa Have Better Domain-General Cognition than others? A Meta-Analysis of Nonhuman Primate Studies," *Evol. Psychol.*, vol. 4, no. 1, p. 147470490600400, Jan. 2006.

[36] M. M. Symington, "Fission-fusion social organization inAteles andPan," *Int. J. Primatol.*, vol. 11, no. 1, pp. 47–61, Feb. 1990.

[37] K. Gupta, K. Deep, and J. C. Bansal, "Spider monkey optimization algorithm for constrained optimization problems," *Soft Comput.*, vol. 21, no. 23, pp. 6933–6962, 2017.

[38] V. Agrawal, R. Rastogi, and D. C. Tiwari, "Spider Monkey Optimization: a survey," *Int. J. Syst. Assur. Eng. Manag.*, vol. 9, no. 4, pp. 929–941, 2018.

[39] A. Sharma, H. Sharma, A. Bhargava, and N. Sharma, "Power law-based local search in spider monkey optimisation for lower order system modelling," *Int. J. Syst. Sci.*, vol. 48, no. 1, pp. 150–160, Jan. 2017.

[40] A. A. Al-Azza, A. A. Al-Jodah, and F. J. Harackiewicz, "Spider Monkey Optimization: A Novel Technique for Antenna Optimization," *IEEE Antennas Wirel. Propag. Lett.*, vol. 15, no. c, pp. 1016–1019, 2016.

[41] R. Cheruku, D. R. Edla, and V. Kuppili, "SM-RuleMiner: Spider monkey based rule miner using novel fitness function for diabetes classification," *Comput. Biol. Med.*, vol. 81, pp. 79–92, Feb. 2017.

[42] J. Dhar and S. Arora, "Designing fuzzy rule base using Spider Monkey Optimization Algorithm in cooperative framework," *Futur. Comput. Informatics J.*, vol. 2, no. 1, pp. 31–38, 2017.

[43] R. Rohilla, V. Sikri, and R. Kapoor, "Spider monkey optimisation assisted particle filter for robust object tracking," *IET Comput. Vis.*, vol. 11, no. 3, pp. 207–219, 2016.

[44] A. Kaur, "Comparison Analysis of CDMA Multiuser Detection using PSO and SMO," *Int. J. Comput. Appl.*, vol. 133, no. 2, pp. 47–50, Jan. 2016.

[45] U. Singh, R. Salgotra, and M. Rattan, "A Novel Binary Spider Monkey Optimization Algorithm for Thinning of Concentric Circular Antenna Arrays," *IETE J. Res.*, vol. 62, no. 6, pp. 736–744, 2016.

[46] B. Omkar, D. Preet, D. Swarada, and D. Poonam, "Dengue Fever classification using SMO Optimization Algorithm," *Int. Res. J. Eng. Technol.*, vol. 4, no. 10, pp. 1683–1686, 2017.

[47] K. Selvam and D. M. V. Kumar, "International journal of renewable energy research IJRER.," *Int. J. Renew. Energy Res.*, vol. 7, no. 1, pp. 146–156, 2017.

[48] A. F. Ali, "An Improved Spider Monkey Optimization for Solving a Convex Economic Dispatch Problem," in *Modeling and Optimization in Science and Technologies*, 2017, pp. 425–448.

[49] M. Ehteram, H. Karami, and S. Farzin, "Reducing Irrigation Deficiencies Based Optimizing Model for Multi-Reservoir Systems Utilizing Spider Monkey Algorithm," *Water Resour.*

Manag., vol. 32, no. 7, pp. 2315–2334, May 2018.

[50] N. Mittal, U. Singh, R. Salgotra, and B. S. Sohi, "A boolean spider monkey optimization based energy efficient clustering approach for WSNs," *Wirel. Networks*, vol. 24, no. 6, pp. 2093–2109, 2018.

[51] P. R. Singh, M. A. Elaziz, and S. Xiong, "Modified Spider Monkey Optimization based on Nelder–Mead method for global optimization," *Expert Syst. Appl.*, vol. 110, pp. 264–289, Nov. 2018.

[52] R. K. Sandeep Kumar, Vivek Kumar Sharma, "Self-Adaptive Spider Monkey Optimization Algorithm for Engineering Optimization Problems," *Int. J. Information, Commun. Comput. Technol.*, vol. 2, no. 2, pp. 96–107, 2014.

[53] S. Kumar, V. K. Sharma, and R. Kumari, "Modified Position Update in Spider Monkey Optimization Algorithm," *Int. J. Emerg. Technol. Comput. Appl. Sci.*, vol. 7, no. April, pp. 198–204, 2014.

[54] S. Kumar, R. Kumari, and V. K. Sharma, "Fitness Based Position Update in Spider Monkey Optimization Algorithm," *Procedia Comput. Sci.*, vol. 62, pp. 442–449, 2015.

[55] A. Sharma, A. Sharma, B. K. Panigrahi, D. Kiran, and R. Kumar, "Ageist Spider Monkey Optimization algorithm," *Swarm Evol. Comput.*, vol. 28, pp. 58–77, 2016.

[56] M. Nagy, Z. Ákos, D. Biro, and T. Vicsek, "Hierarchical group dynamics in pigeon flocks," *Nature*, vol. 464, no. 7290, pp. 890–893, Apr. 2010.

[57] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, no. 7025, pp. 513–516, Feb. 2005.

[58] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.

[59] F. B. Ozsoydan and A. Baykasoglu, "A multi-population firefly algorithm for dynamic optimization problems," in *2015 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 2015, pp. 1–7.

[60] F. B. Ozsoydan and A. Baykasoğlu, "Quantum firefly swarms for multimodal dynamic optimization problems," *Expert Syst. Appl.*, 2019.

[61] A. Schrijver, "On the History of Combinatorial Optimization (Till 1960)," *Handbooks in Operations Research and Management Science*. 2005.

[62] S. Ahn, S. Lee, and H. Bahn, "A smart elevator scheduler that considers dynamic changes of energy cost and user traffic," *Integr. Comput. Aided. Eng.*, vol. 24, no. 2, pp. 187–202, Mar. 2017.

[63] S. A. Bagloee, M. Sarvi, M. Patriksson, and M. Asadi, "Optimization for Roads' Construction: Selection, Prioritization, and Scheduling," *Comput. Civ. Infrastruct. Eng.*, vol. 33, no. 10, pp. 833–848, Oct. 2018.

[64] M. Alinizzi, S. Chen, S. Labi, and A. Kandil, "A Methodology to Account for One-Way Infrastructure Interdependency in Preservation Activity Scheduling," *Comput. Civ. Infrastruct. Eng.*, vol. 33, no. 11, pp. 905–925, Nov. 2018.

[65] S. Xie, C. Lei, and Y. Ouyang, "A Customized Hybrid Approach to Infrastructure Maintenance Scheduling in Railroad Networks under Variable Productivities," *Comput. Civ. Infrastruct. Eng.*, vol. 33, no. 10, pp. 815–832, Oct. 2018.

[66] J. D. García-Nieves, J. L. Ponz-Tienda, A. Salcedo-Bernal, and E. Pellicer, "The Multimode

Resource-Constrained Project Scheduling Problem for Repetitive Activities in Construction Projects," *Comput. Civ. Infrastruct. Eng.*, vol. 33, no. 8, pp. 655–671, Aug. 2018.

[67] T.-Y. Liao, "On-Line Vehicle Routing Problems for Carbon Emissions Reduction," *Comput. Civ. Infrastruct. Eng.*, vol. 32, no. 12, pp. 1047–1063, Dec. 2017.

[68] V. Zverovich, L. Mahdjoubi, P. Boguslawski, and F. Fadli, "Analytic Prioritization of Indoor Routes for Search and Rescue Operations in Hazardous Environments," *Comput. Civ. Infrastruct. Eng.*, vol. 32, no. 9, pp. 727–747, Sep. 2017.

[69] K. L. Hoffman, M. Padberg, and G. Rinaldi, "Traveling Salesman Problem," in *Encyclopedia of Operations Research and Management Science*, Boston, MA: Springer US, 2013, pp. 1573–1578.

[70] N. Siddique and H. Adeli, "Nature Inspired Computing: An Overview and Some Future Directions," *Cognit. Comput.*, vol. 7, no. 6, pp. 706–714, Dec. 2015.

[71] N. H. Siddique and H. Adeli, *Nature-Inspired Computing: Physics and Chemistry-Based Algorithms*. CRC Press/Taylor & Francis Group, 2017.

[72] J. Grefenstette, R. Gopal, B. Roamaita, and D. Van Gucht, "Evolutionary Computation and the Traveling Salesman Problem," in *Evolutionary Computation*, IEEE, 2009, pp. 523–540.

[73] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artif. Intell. Rev.*, vol. 13, no. 2, pp. 129–170, 1999.

[74] J. Y. Potvin, "Genetic Algorithms for Travelling Salesman Problem," *Ann. Oper. Res.*, vol. 63, pp. 339–370, 1996.

[75] N. Siddique and H. Adeli, "Simulated Annealing, Its Variants and Engineering Applications," *Int. J. Artif. Intell. Tools*, vol. 25, no. 06, p. 1630001, Dec. 2016.

[76] P. J. M. V. L. E. H. L. Aarts, J. H. M. Korst, "A Quantitative Analysis of Simulated Annealing Algorithm: A Case Study for Travelling Salesman problem," *J. Stat. Phys.*, vol. 50, pp. 189–206, 1988.

[77] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *J. Optim. Theory Appl.*, vol. 45, no. 1, pp. 41–51, Jan. 1985.

[78] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao, "Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search," *Appl. Soft Comput.*, vol. 11, no. 4, pp. 3680–3689, Jun. 2011.

[79] N. Siddique and H. Adeli, "Physics-based search and optimization: Inspirations from nature," *Expert Syst.*, vol. 33, no. 6, pp. 607–623, Dec. 2016.

[80] N. Javadian, M. Gol Alikhani, and R. Tavakkoli-Moghaddam, "A discrete binary version of the electromagnetism-like heuristic for solving traveling salesman problem," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008.

[81] N. Siddique and H. Adeli, "Gravitational Search Algorithm and Its Variants," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 30, no. 08, p. 1639001, Sep. 2016.

[82] N. Siddique and H. Adeli, "Applications of Gravitational Search Algorithm in Engineering," *J. Civ. Eng. Manag.*, vol. 22, no. 8, pp. 981–990, Nov. 2016.

[83] M. B. Dowlatshahi, H. Nezamabadi-pour, and M. Mashinchi, "A discrete gravitational search algorithm for solving combinatorial optimization problems," *Inf. Sci. (Ny).*, vol. 258, pp. 94–107, Feb. 2014.

[84] A. Montero, I. Méndez-Díaz, and J. J. Miranda-Bront, “An integer programming approach for the time-dependent traveling salesman problem with time windows,” *Comput. Oper. Res.*, vol. 88, pp. 280–289, Dec. 2017.

[85] Y. Salii, “Revisiting dynamic programming for precedence-constrained traveling salesman problem and its time-dependent generalization,” *Eur. J. Oper. Res.*, vol. 272, no. 1, pp. 32–42, Jan. 2019.

[86] E. Bonabeau, M. Dorigo, and G. Theraulaz, “Inspiration for optimization from social insect behaviour,” *Nature*, vol. 406, no. 6791, pp. 39–42, Jul. 2000.

[87] M. Dorigo and L. M. Gambardella, “Ant colonies for the travelling salesman problem,” *Biosystems*, vol. 43, no. 2, pp. 73–81, Jul. 1997.

[88] J. B. Escario, J. F. Jimenez, and J. M. Giron-Sierra, “Ant colony extended: Experiments on the travelling salesman problem,” *Expert Syst. Appl.*, vol. 42, no. 1, pp. 390–410, 2015.

[89] J. Yang, X. Shi, M. Marchese, and Y. Liang, “An ant colony optimization method for generalized TSP problem,” *Prog. Nat. Sci.*, vol. 18, no. 11, pp. 1417–1422, Nov. 2008.

[90] M. A. H. Akhand, S. Akter, and M. A. Rashid, “Velocity Tentative Particle Swarm Optimization to solve TSP,” in *2013 International Conference on Electrical Information and Communication Technology (EICT)*, 2014, pp. 1–6.

[91] Kang-Ping Wang, Lan Huang, Chun-Guang Zhou, and Wei Pang, “Particle swarm optimization for traveling salesman problem,” in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, 2003, pp. 1583–1585.

[92] J. Zhang and W. Si, “Improved Enhanced Self-Tentative PSO algorithm for TSP,” in *2010 Sixth International Conference on Natural Computation*, 2010, pp. 2638–2641.

[93] V. Pandiri and A. Singh, “A hyper-heuristic based artificial bee colony algorithm for k -Interconnected multi-depot multi-traveling salesman problem,” *Inf. Sci. (Ny).*, vol. 463–464, pp. 261–281, Oct. 2018.

[94] “TSPLIB – library of sample symmetric Traveling Salesman Problem instance.,” 1995. [Online]. Available: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/.

[95] J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.

[96] C. Zaiontz, “Wilcoxon Signed-Ranks Test,” *Real Statistics Using Excel*. [Online]. Available: http://www.real-statistics.com/non-parametric-tests/wilcoxon-signed-ranks-test/.